What is claimed is:

1. A method of simulating a logic design, the method comprising:

storing a first state to identify in a simulation of a logic design whether a node included in the logic design has a logic high value;

storing a second state to identify in simulation of the logic design whether the node has a logic low value;

storing a third state to identify in simulation of the logic design whether the node has an undefined state; and

determining an output of the node in simulation of the logic design based on the first state, the second state, and the third state.

2. The method of claim 1 further comprising determining whether the output of the node in simulation of the logic design has a high impedance state based on the first state, the second state, and the third state.

3. The method of claim 1 further comprising storing the third state only if at least two sources drive the node.

4. The method of claim 1 further comprising:

performing four state simulation of the logic design

subsequent to the simulation of the logic design if the simulation of the logic design was successful.

5. The method of claim 1 further comprising:

determining the output based on which of the first state, the second state, and the third state is at a logic high.

6. An article for simulating a logic design, the article comprising:

a machine-readable medium which contains machine-executable instructions, the instructions causing a machine to:

store a first state to identify in a simulation of a logic design whether a node included in the logic design has a logic high value;

store a second state to identify in simulation of the logic design whether the node has a logic low value;

store a third state to identify in simulation of the logic design whether the node has an undefined state; and

determine an output of the node in simulation of the logic design based on the first state, the second state, and the third state.

7. The article of claim 6 further causing a machine to determine whether the output of the node in simulation of the

logic design has a high impedance state based on the first state, the second state, and the third state.

8. The article of claim 6 further causing a machine to store the third state only if at least two sources drive the node.

9. The article of claim 6 further causing a machine to:

perform four state simulation of the logic design subsequent to the simulation of the logic design if the simulation of the logic design was successful.

10. The article of claim 6 further causing a machine to:

determine the output based on which of the first state, the second state, and the third state is at a logic high.

11. An apparatus for simulating a logic design, the apparatus comprising:

a memory that stores executable instructions; and

a processor that executes the instructions to:

store a first state to identify in a simulation of a logic design whether a node included in the logic design has a logic high value,

store a second state to identify in simulation of the logic design whether the node has a logic low value,

store a third state to identify in simulation of the

logic design whether the node has an undefined state, and

determine an output of the node in simulation of the logic design based on the first state, the second state, and the third state.

12. The apparatus of claim 11 further executing the instructions to determine whether the output of the node in simulation of the logic design has a high impedance state based on the first state, the second state, and the third state.

13. The apparatus of claim 11 further executing the instructions to store the third state only if at least two sources drive the node.

14. The apparatus of claim 11 further executing the instructions to perform four state simulation of the logic design subsequent to the simulation of the logic design if the simulation of the logic design was successful.

15. The apparatus of claim 11 further executing the instructions to determine the output based on which of the first state, the second state, and the third state is at a logic high.

16. A method of simulating a logic design, the method comprising:

storing only three bits of state information for a node included in a logic design; and

checking the three bits in simulating operation of the logic design.

17. The method of claim 16 further comprising determining an output of the node in simulation of the logic design based on the three bits of state information.

18. The method of claim 16 further comprising determining whether an output of the node in simulation of the logic design has a logic high state, a logic low state, or an undefined state based the three bits of state information.

19. The method of claim 16 further comprising determining whether an output of the node in simulation of the logic design has a high impedance state based the three bits of state information.

20. The method of claim 16 further comprising storing four bits of state information for the node included in the logic design;

checking the four bits in a second simulating operation of the logic design; and

determining whether an output of the node in simulation of the logic design has a logic high state, a logic low state,

an undefined state, or a high impedance state based the four bits of state information.

21. The method of claim 16 further comprising determining an output of the node based on which of the three bits of information is at a logic high.

22. A method for simulating a logic design using cycle-based simulation, the method comprising:

attempting to write a result of a logic computation instruction to a location in a write-protected memory page storing an original value;

copying the instruction in a second memory page;

executing the second memory page starting with the instruction;

unprotecting the write-protected memory page and storing a result of the instruction at the location in the write-protected memory page;

rewriting the original value to the location in the write-protected memory page if the original value differs from the result; and

re-protecting the write-protected memory page.

23. The method of claim 22 further comprising inserting an illegal instruction after the instruction in the second memory page that triggers the rewriting and the re-protecting.

24. The method of claim 22 further comprising generating an exception if a computer system attempts to write the result of the logic computation instruction to the location in the write-protected memory page.